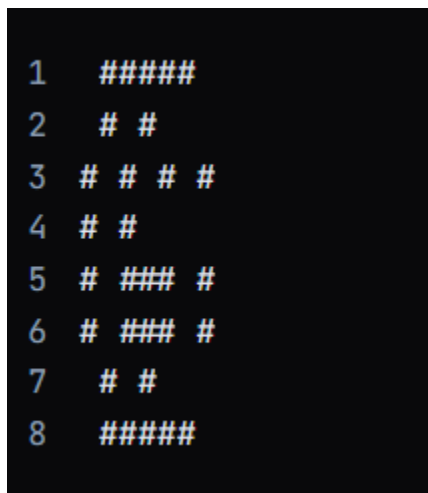


JFo Section 2 Practice

ProblemSet2_1

```
public class ProblemSet2_1 {  
    public static void main(String[] args) {  
        // Use 8 print statements to print a smiley face.  
        // The art will rely on only a single character, besides space, such as X or #.  
  
        System.out.println("" ##### "");  
        System.out.println("" # # "");  
        System.out.println(""# # # #"");  
        System.out.println(""# #"");  
        System.out.println(""# ### #"");  
        System.out.println(""# ### #"");  
        System.out.println("" # # "");  
        System.out.println("" ##### "");  
    }  
}
```

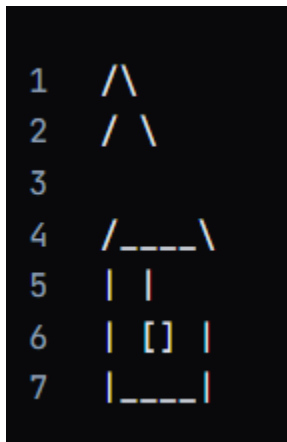


Problem set2_2

ASCII art is a graphic design technique that uses printable characters from the ASCII standard to create images and designs. ASCII (American Standard Code for Information Interchange)

includes 128 characters, such as letters, numbers, punctuation marks, and control characters. By carefully arranging these characters, artists can create detailed images, patterns, and representations of objects.

```
public class ProblemSet2_2 {  
    public static void main(String[] args) {  
        // This ASCII art depicts a simple house with a roof, a door, and windows.  
        System.out.println("" /\ "");  
        System.out.println("" / \ "");  
  
        System.out.println("" /____\ "");  
        System.out.println("" | | "");  
        System.out.println("" | [] | "");  
        System.out.println("" |____| "");  
    }  
}
```



JF section 5 Practice

```
import java.util.Scanner;  
  
public class ColorRange {  
    public static void main(String[] args) {  
        // Create a Scanner object for user input  
        Scanner scanner = new Scanner(System.in);
```

```

// Define the valid range for each color component
int minRange = 0;
int maxRange = 255;

// Prompt user to enter RGB values
System.out.print("<div data-bbox=">Enter the Red component (0-255): </div>");
int red = scanner.nextInt();

System.out.print("<div data-bbox=">Enter the Green component (0-255): </div>");
int green = scanner.nextInt();

System.out.print("<div data-bbox=">Enter the Blue component (0-255): </div>");
int blue = scanner.nextInt();

// Check if the RGB values are within the specified range
boolean isValidRed = red >= minRange && red <= maxRange;
boolean isValidGreen = green >= minRange && green <= maxRange;
boolean isValidBlue = blue >= minRange && blue <= maxRange;

// Display results
System.out.println("<div data-bbox=">\nColor Component Validity:</div>");
System.out.println("<div data-bbox=">Red: </div> + (isValidRed ? "<div data-bbox=">Valid</div>" : "<div data-bbox=">Invalid</div>"));
System.out.println("<div data-bbox=">Green: </div> + (isValidGreen ? "<div data-bbox=">Valid</div>" : "<div data-bbox=">Invalid</div>"));
System.out.println("<div data-bbox=">Blue: </div> + (isValidBlue ? "<div data-bbox=">Valid</div>" : "<div data-bbox=">Invalid</div>"));

// Close the scanner
scanner.close();
}
}

```

2. To build a TrafficLightChecker class, you should focus on creating a system that simulates the behavior of traffic lights. This simulation can be expanded to include functionalities such as checking the current light

status, determining the duration of each light phase, and providing a mechanism for switching between lights.

```
import java.util.Scanner;

public class TrafficLightChecker {

    // Enum to define traffic light states

    private enum TrafficLight {

        RED, YELLOW, GREEN

    }

    // Method to get the next traffic light based on current light

    private static TrafficLight getNextLight(TrafficLight current) {

        switch (current) {

            case RED:

                return TrafficLight.GREEN;

            case YELLOW:

                return TrafficLight.RED;

            case GREEN:

                return TrafficLight.YELLOW;

            default:

                throw new IllegalArgumentException(""Unexpected value: " +

                    current);

        }

    }

    // Method to display the traffic light status

    private static void displayStatus(TrafficLight light) {

        switch (light) {

            case RED:

                System.out.println(""The light is RED. Please stop."");

                break;

            case YELLOW:

                System.out.println(""The light is YELLOW. Prepare to stop."");

                break;
```

```

case GREEN:

System.out.println(""The light is GREEN. You may go."");

break;

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt user for the initial traffic light state

System.out.print(""Enter the current traffic light color (RED,
YELLOW, GREEN): "");

String input = scanner.next().toUpperCase();

TrafficLight currentLight;

try {

// Convert the input string to TrafficLight enum
currentLight = TrafficLight.valueOf(input);

} catch (IllegalArgumentException e) {

System.out.println(""Invalid color entered. Please enter RED,
YELLOW, or GREEN."");

scanner.close();

return;

}

// Display the current light status
displayStatus(currentLight);

// Determine the next traffic light state
TrafficLight nextLight = getNextLight(currentLight);

// Display the next light status

System.out.println(""The next light will be: " + nextLight);

displayStatus(nextLight);

```

```
// Close the scanner
scanner.close();
}
}
```

```
1 Enter the current traffic light color (RED, YELLOW, GREEN): RED
2 The light is RED. Please stop.
3 The next light will be: GREEN
4 The light is GREEN. You may go.
```

3. To implement a `TrafficLightSwitch` class that simulates switching traffic lights, you might want to create functionality for managing the current state of the traffic light, switching between states, and possibly displaying information about the light. Below is a detailed example of how you could set up this class:

```
import java.util.Scanner;

public class TrafficLightSwitch {

    // Enum to define traffic light states
    private enum TrafficLight {
        RED, YELLOW, GREEN
    }

    // Method to get the next traffic light based on current light
    private static TrafficLight getNextLight(TrafficLight current) {
        switch (current) {
            case RED:
                return TrafficLight.GREEN;
            case YELLOW:
                return TrafficLight.RED;
            case GREEN:
                return TrafficLight.YELLOW;
        }
    }
}
```

default:

```
throw new IllegalArgumentException(""Unexpected value: " + current);
```

```
}
```

```
}
```

```
// Method to display the traffic light status
```

```
private static void displayStatus(TrafficLight light) {
```

```
switch (light) {
```

```
case RED:
```

```
System.out.println(""The light is RED. Please stop.");
```

```
break;
```

```
case YELLOW:
```

```
System.out.println(""The light is YELLOW. Prepare to stop.");
```

```
break;
```

```
case GREEN:
```

```
System.out.println(""The light is GREEN. You may go.");
```

```
break;
```

```
}
```

```
}
```

```
public static void main(String[] args) {
```

```
Scanner scanner = new Scanner(System.in);
```

```
// Prompt user for the initial traffic light state
```

```
System.out.print(""Enter the current traffic light color (RED, YELLOW,  
GREEN): ");
```

```
String input = scanner.next().toUpperCase();
```

```
TrafficLight currentLight;
```

```
try {
```

```
// Convert the input string to TrafficLight enum
```

```
currentLight = TrafficLight.valueOf(input);
```

```

    } catch (IllegalArgumentException e) {
        System.out.println(""Invalid color entered. Please enter RED, YELLOW, or
        GREEN.");
        scanner.close();
        return;
    }
    // Display the current light status
    displayStatus(currentLight);
    // Determine the next traffic light state
    TrafficLight nextLight = getNextLight(currentLight);
    // Display the next light status
    System.out.println(""The next light will be: " + nextLight);
    displayStatus(nextLight);
    // Close the scanner

    scanner.close();
}
}

```

```

1 Enter the current traffic light color (RED, YELLOW, GREEN): BLUE
2 Invalid color entered. Please enter RED, YELLOW, or GREEN.

```

JF section 4 practice

To create a class `ComputeMethods` that utilizes the `java.util.Random` class, you might want to implement methods that perform various computations or generate random data. Below are some examples of what you can include in this class:

Example 1: Generate Random Numbers and Basic Computations

1. Generating Random Integers and Doubles:

- o Methods to generate random integers within a range.

- o Methods to generate random doubles within a range.

2. Computations Using Random Numbers:

- o Methods to compute the sum, average, or other statistics using generated random numbers.

Here's a complete example of the ComputeMethods class:

```
import java.util.Random;

public class ComputeMethods {
    private Random random;

    public ComputeMethods() {
        // Initialize the Random object
        random = new Random();
    }

    // Method to generate a random integer between min and max
    (inclusive)
    public int getRandomInt(int min, int max) {

        return random.nextInt((max - min) + 1) + min;
    }

    // Method to generate a random double between min and max
    public double getRandomDouble(double min, double max) {
        return min + (max - min) * random.nextDouble();
    }

    // Method to compute the average of an array of integers
    public double computeAverage(int[] numbers) {
        if (numbers.length == 0) return 0;
        int sum = 0;
        for (int number : numbers) {
```

```
sum += number;
}
return (double) sum / numbers.length;
}
```

```
// Method to compute the sum of an array of doubles
```

```
public double computeSum(double[] numbers) {
    double sum = 0.0;
    for (double number : numbers) {
        sum += number;
    }
}
```

```
return sum;
}
```

```
// Method to generate an array of random integers
```

```
public int[] generateRandomIntArray(int size, int min, int max) {
    int[] array = new int[size];
    for (int i = 0; i < size; i++) {
        array[i] = getRandomInt(min, max);
    }
    return array;
}
```

```
// Method to generate an array of random doubles
```

```
public double[] generateRandomDoubleArray(int size, double min,
    double max) {
    double[] array = new double[size];
    for (int i = 0; i < size; i++) {
        array[i] = getRandomDouble(min, max);
    }
}
```

```

}
return array;
}
public static void main(String[] args) {
    ComputeMethods cm = new ComputeMethods();
    // Generate random numbers and compute results

    int[] intArray = cm.generateRandomIntArray(5, 1, 100);
    double[] doubleArray = cm.generateRandomDoubleArray(5, 0.0,
    1.0);
    System.out.println(""Random Integers:"");
    for (int num : intArray) {
        System.out.print(num + "" ");
    }
    System.out.println(""\nAverage of Integers: " +
    cm.computeAverage(intArray));
    System.out.println(""\nRandom Doubles:"");
    for (double num : doubleArray) {
        System.out.print(num + "" ");
    }
    System.out.println(""\nSum of Doubles: " +
    cm.computeSum(doubleArray));
}
}

```

```

1 Random Integers:
2 14 73 28 41 91
3 Average of Integers: 49.4
4
5 Random Doubles:
6 0.1319218144545251 0.9354404345716553 0.06324555321255111 0.292482993655441
7 Sum of Doubles: 3.241021500569892

```

JF Section 3 Practice

The JOptionPane class from the javax.swing package is used to create simple dialogs for user interaction in Java. In your JavaLibsPractice class, you’ve started by prompting the user to enter their name. Here’s a complete example demonstrating how to use JOptionPane to interact with the user, including displaying a message box and handling user input.

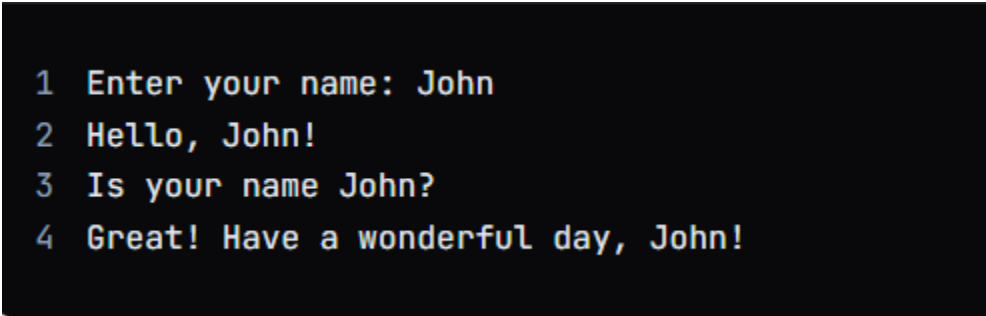
```

import javax.swing.JOptionPane;

public class JavaLibsPractice {
    public static void main(String[] args) {
        // Prompt the user to enter their name
        String name = JOptionPane.showInputDialog("Enter your name:");
        // If the user does not enter a name, handle it gracefully
        if (name == null || name.isEmpty()) {
            name = "Guest";
        }
        // Display a personalized greeting message
        JOptionPane.showMessageDialog(null, "Hello, " + name + "!");
        // Optional: Show the user's name in a confirmation dialog
        int response = JOptionPane.showConfirmDialog(null, "Is your name " + name
            + "?", "Name Confirmation", JOptionPane.YES_NO_OPTION);
        if (response == JOptionPane.YES_OPTION) {
            JOptionPane.showMessageDialog(null, "Great! Have a wonderful day, " +

```

```
name + &quot;!&quot;);  
} else {  
JOptionPane.showMessageDialog(null, &quot;Oops! Please restart the  
application and try again.&quot;);  
}  
}  
}
```



```
1 Enter your name: John  
2 Hello, John!  
3 Is your name John?  
4 Great! Have a wonderful day, John!
```